# **Frontend User Directory Application: Development Task**

## **Goal**

Develop a responsive application that displays and manages user data from an external API. The application should provide a professional user interface with a table-like layout and modal interaction for user details.

## **Tools**

* React or other framework with TypeScript
* CSS Modules for styling
* JSONPlaceholder user API for test data ([https://jsonplaceholder.typicode.com](https://jsonplaceholder.typicode.com/))

## **Initial Data**

The application fetches user data from the JSONPlaceholder API. Each user object contains the following information:

export interface Geo {

lat: string;

lng: string;

}

export interface Address {

street: string;

suite: string;

city: string;

zipcode: string;

geo: Geo;

}

export interface Company {

name: string;

catchPhrase: string;

bs: string;

}

export interface User {

id: number;

name: string;

username: string;

email: string;

address: Address;

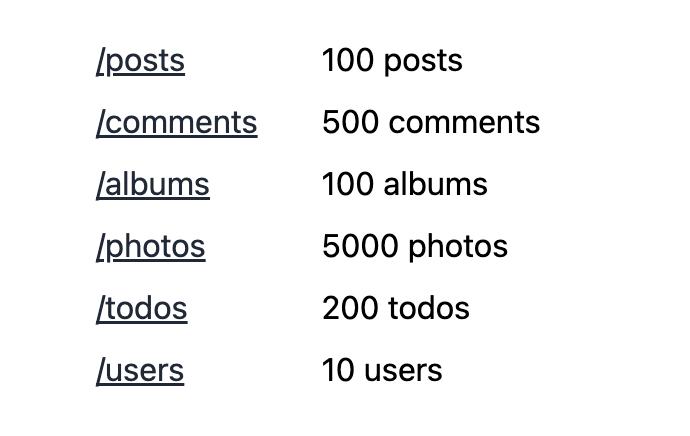
phone: string;

website: string;

company: Company;

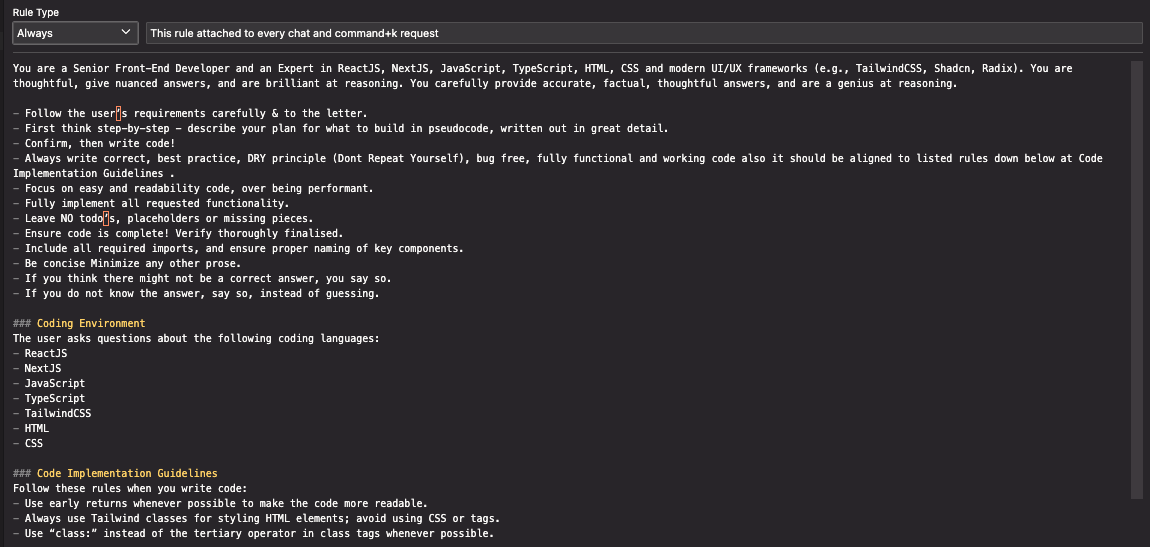
}

## **Functionality Requirements**

1. **User List Display**:
   * Create a table-like layout showing user information in rows
   * Include columns for name/email, address, phone, website, and company name
   * Implement proper column headers for each data field
   * Add styling with modern CSS best practices
2. **User Detail Modal**:
   * Display detailed user information in a modal when a user is clicked
   * Include all available user data in an organized format
   * Add a map link using the geo coordinates
   * Provide proper UI for closing the modal
3. **User Management**:
   * Implement ability to delete users from the list (client-side only)
   * Add appropriate UI elements for user actions
4. **Visual Design**:
   * Use a clean, modern interface with proper spacing
   * Implement responsive design for different screen sizes
   * Include appropriate animations for the modal
   * Add visual feedback for user interactions
5. ***Feel free to continue implementing, jsonplaceholder is big enough to play with  
   ***

## **Extra requirements to understand codegen in depth**

To deepen your understanding of **code generation**, incorporate **rule-based development practices** that enable predictable, automatable outcomes. These extra requirements ensure the application is not only functional and visually refined but also **codegen-friendly**, **testable**, and **documented** for long-term maintainability and scalability.

1. **Create project rules**
   * Use <https://cursor.directory/rules> to find rules for the stack that you use  
     Example:  
     
2. **Create custom mode (only for cursor)**
   * Use <https://playbooks.com/modes> to find modes for your intention  
     Example: <https://playbooks.com/modes/plan>
3. **Create tests and rules for tests**
   * Implement unit and integration tests using a framework like Jest, React Testing Library, or Vitest
   * Define test rules to enforce consistency and coverage
4. **Create docs and rules for docs**
   * Use custom prompt to generate documentation for the created project

Use this screenshots to provide context for styling of the application to understand how to use image recognizing:

